Job DSL Plugin

Plugin Information
View Job DSL on the plugin site for more information.

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- Permission check bypass for access and modification of Jenkins model objects
- Arbitrary code execution vulnerability
- Script security sandbox bypass

The job-dsl-plugin allows the programmatic creation of projects using a DSL. Pushing job creation into a script allows you to automate and standardize your Jenkins installation, unlike anything possible before.

Summary

Jenkins is a wonderful system for managing builds, and people love using its UI to configure jobs. Unfortunately, as the number of jobs grows, maintaining them becomes tedious, and the paradigm of using a UI falls apart. Additionally, the common pattern in this situation is to copy jobs to create new ones, these "children" have a habit of diverging from their original "template" and consequently it becomes difficult to maintain consistency between these jobs.

The Jenkins job-dsl-plugin attempts to solve this problem by allowing jobs to be defined with the absolute minimum necessary in a programmatic form, with the help of templates that are synced with the generated jobs. The goal is for your project to be able to define all the jobs they want to be related to their project, declaring their intent for the jobs, leaving the common stuff up to a template that were defined earlier or hidden behind the DSL.

A bulk of the documentation for this plugin is in the GitHub wiki

Please join the mailing list (https://groups.google.com/d/forum/job-dsl-plugin) for any user or developer questions.

Configuration

After installing the plugin, you'll get a new Build Step entry named "Process Job DSLs". Type the DSL directly in the text box, or point to a file in the workspace.

Basics

The DSL allows the definition of a job, and then offers a useful set of functions to configure common Jenkins items. A *configure* is available to give direct access to the *config.xml* before generating the job. The script is groovy code, which can be very powerful. Here's an example to create a job for each branch in a git repo:

```
def project = 'quidryan/aws-sdk-test'
def branchApi = new URL("https://api.github.com/repos/${project}/branches")
def branches = new groovy.json.JsonSlurper().parse(branchApi.newReader())
branches.each {
    def branchName = it.name
    def jobName = "${project}-${branchName}".replaceAll('/','-')
    job(jobName) {
        scm {
            git("git://github.com/${project}.git", branchName)
        }
        steps {
            maven("test -Dproject.name=${project}/${branchName}")
        }
    }
}
```

Features

- DSL - Scriptable via Groovy

- DSL - Direct control of XML, so that anything possible in a config.xml is possible via the DSL
- DSL - Helper methods for common job configurations, e.g. scm, triggers, build steps
- Plugin - DSL can be put directly in a job
- Plugin - DSL can be put into SCM and polled using standard SCM triggering
- Plugin - Multiple DSLs can be referenced at a time
- Plugin - Tracks Templates used, will update derivative jobs when template is changed

Usage

See the wiki for specific steps and other examples.

1. Create your set of Jenkins jobs which will serve as the templates (It is a good idea to use a naming convention for these jobs which clearly indicates that these are templates)
2. Create a Jenkins Job using the Free-style project style to run your DSL Scripts. This is called a "Seed" job.
3. Configure the seed job, by adding a "Build Step" of type "Process Job DSLs" and paste in the body of the DSL.
4. Run the seed to generate your new jobs from your script. When successful, and the "build result" page will list the jobs which have been successfully created.
5. Finally, it is good practice to organise your Jenkins UI with some new tabs so that the management and template jobs are not the first thing a user sees when they login

The tutorial provides explicit steps to use the plugin.

Roadmap

- Gradle plugin, for creation of jobs from Gradle
- Additional methods to cover the most common configuration items.

About

This plugin is started as a hack-a-thon at the Java Posse Roundup#457. It was primarily developed for use at Netflix, but is now used at a wide range of companies and projects. It is licensed under Apache License, Version 2.0.

Changelog

See Release Notes in the github wiki.